Twirre: Architecture for autonomous mini-UAVs using interchangeable commodity components

J. van de Loosdrecht*, K. Dijkstra, J.H. Postma, W. Keuning and D. Bruin
NHL University of Applied Sciences, Center of Expertise Computer Vision,
P.O. Box 1080, 8900 CB, Leeuwarden, The Netherlands

ABSTRACT

Twirre is a new architecture for mini-UAV platforms designed for autonomous flight in both GPS-enabled and GPS-deprived applications. The architecture consists of low-cost hardware and software components. High-level control software enables autonomous operation. Exchanging or upgrading hardware components is straightforward and the architecture is an excellent starting point for building low-cost autonomous mini-UAVs for a variety of applications. Experiments with an implementation of the architecture are in development, and preliminary results demonstrate accurate indoor navigation.

1 INTRODUCTION

Nowadays, there is increasing interest in using mini unmanned aerial vehicles (UAVs) in a multitude of applications. For example, in the Netherlands, UAVs can be used for the inspection of agricultural lands, reviewing annual ditch cleanings, or the inspection of wind turbine blades. The market of UAVs is fast-growing and new hardware, software and applications quickly emerge. UAVs with cameras mounted on them can already be purchased as commodities.

Currently, however, in most applications the UAVs are controlled manually and the images are inspected by humans. Performing these tasks autonomously would make them less labor-intensive and thus more cost-effective. This involves autonomous flying of UAVs and automated (real-time) processing of recorded images to generate process or business support information.

This article focuses on such autonomous flight. Autopilots using GPS navigation are already a commodity [1, 2, 3], though autonomous systems for GPS-deprived environments are not. This is because complex sensor fusion and image analysis are needed to be able to cancel all dynamic effects during flight and to estimate the UAV's position in 3D. There are many UAV platforms available, often expensive and based on dedicated custom-built hardware and software.

In this paper the Twirre architecture will be introduced, which is a low-cost UAV platform designed for autonomous flight. The platform should be able to perform missions autonomously from takeoff to landing, without assistance of manual control. The intention is to submit the system software to the public domain. To make the platform attractive for both experimental and commercial use, the following four design requirements have been kept in mind.

1. The platform consists of low-cost components. Many applications require costs to be minimized. A UAV will likely have a broader application range if the cost of the components are low. Furthermore, using autonomous UAVs always comes with a risk of material damage. For these reasons the UAV platform is composed of low-cost commodity mass-produced components, which can easily be exchanged, repaired or upgraded. Components like flight controllers and GPS modules can be purchased off-the-shelf.

2. The platform is upgradable and extendable. New versions of radio controlled (RC) hardware components and standard processor boards quickly become available. To allow for easy incorporation of new technology in the UAV, components should be upgraded or extended with ease. All software is developed in C(++) and therefore easily portable.

3. The platform is useful in multiple applications. For a UAV platform to be useful in multiple applications it should be flexible. This means that it should be possible to perform both GPS-enabled and GPS-deprived applications. For example, automated imaging of agricultural fields can be done with GPS-enabled flight. However, for inspecting wind turbine blades GPS is not accurate enough and information from additional sensors should be used to estimate the UAV's position. In order to process and interpret this information (including images) in real-time, without the aid of external resources, processing power is required on-board the UAV.

4. The platform should be able to switch instantly and reliably between manual and autonomous control. Regardless the application, safety is of great importance for autonomous flight. Therefore, in autonomous flight, a reliable method for reinstating manual control must be devised.

This paper starts by presenting a short overview of related work in Section 2, after which the Twirre architecture is introduced in Section 3. While the architecture can potentially

*Email address: j.van.de.loosdrecht@nhl.nl

http://www.imavs.org/

http://www.imavs.org/

be used for any type of UAV, Section 4 focuses on an example implementation on a hexacopter which can be used in GPS-deprived environments. Section 5 then shows the layout of several experiments that have been performed with this implementation. They are used to test the architecture. The results and their interpretation are given in Section 6. Section 7 contains a list of conclusions based on the implementation and the experimental results. The paper is wrapped up with a short elaboration on work still to be done in Section 8.

2 RELATED WORK

Shen et al. [4] describe an architecture similar to Twirre. However, Twirre focuses on using low-cost interchangeable commodity components, which is not a design requirement in their paper.

Most other research is focused on small aspects of autonomous flight, like landing [5], planning [6], or integration with specific hardware [7]. More complex tasks are often demonstrated in simulated environments only [8].

The Twirre architecture connects available hardware and software in a modular design, thereby filling the gap between technology and real-world autonomous applications. Research has been performed on popular components which can serve as an implementation for low-level and high-level control of the Twirre architecture. Low-level control encompasses basic flight control, whereas high-level control consists of intelligent navigation.

Low-level control (leveling, maintaining altitude) For low-level control a flight controller is used. Such hardware and software components can be purchased off-the-shelf. Currently, for multicopters, the DJI Naza [1] is a popular choice among RC model pilots. The Paparazzi autopilot is an open source alternative which is mainly used in academic research [9].

The Paparazzi platform contains a safety pilot feature built into the software of the autonomous controller which can be used to switch to manual flight. However, the autonomous controller cannot be bypassed and thus there is no 'galvanic' (in hardware) separation between the RC control and the actuators. Although thoroughly tested, this architecture could potentially lead to unsafe situations.

High-level control (POI interaction) High-level control is used to locate and interact with a *point of interest* (POI, sometimes referred to as *landmark*). Using GPS is one solution for this type of control. A possible POI could then be a GPS location. However, low-cost GPS in itself is only accurate up to a few meters which limits the possible applications. The proposed solution for augmenting position information uses differential GPS [10] or the placement of artificial landmarks in the surroundings [11, 12]. Even though for research purposes artificial landmarks can be useful, most actual applications revolve around markerless navigation.

When relying solely on image analysis in GPS-deprived applications, several control solutions are available. One of them is to use feature detection [4] or the scale invariant feature transform (SIFT), possibly combined with an omnidirectional camera [13], as used with unmanned ground vehicles (UGVs). While UGVs are stable and usually only navigate on a planar surface, UAVs are unstable and navigate in a 3D environment. For a UAV to keep a constant position, a POI on the ground could be used. The drift can then be calculated by analyzing two consecutive frames by an optical flow algorithm [14], which uses a pixel descriptor to detect where points have drifted to. Other solutions are to use laser odometry [4] or simultaneous localization and mapping (SLAM) [15, 16]. However, high-level controllers are generally computationally expensive, which is why UAVs often rely on wireless connections to an external computing station [17]. Since a platform which is independent of external resources is more flexible, the Twirre architecture uses on-board processing.

3 MATERIALS

In this section the components comprising the Twirre architecture are described. The first subsection introduces the general components in an abstract manner, after which the components making up an actual implementation of the platform based on a hexacopter are introduced.

3.1 The Twirre architecture

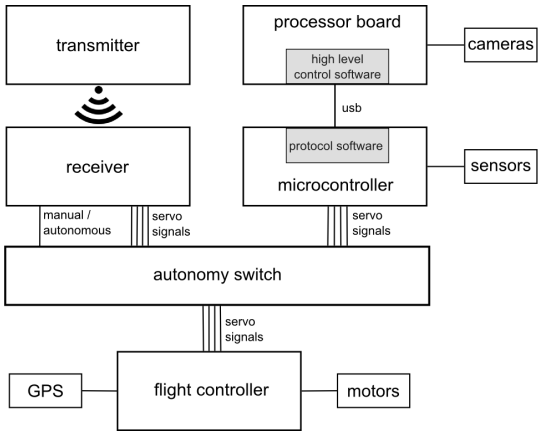

Figure 1: Twirre architecture.

Figure 1 shows the Twirre architecture consisting of six main components and several peripherals. There are two control pipelines. The left hand side shows manual control, while on the right hand side autonomous control is shown. The autonomy switch can be used to instantly switch from the autonomous pipeline to the manual pipeline. At the bottom of Figure 1 a stock flight controller is utilized for low-level con-

[1] http://www.dji.com

http://www.imavs.org/

trol in both manual and autonomous control, to provide the convenient flight control typically used by hobby RC UAVs. The flight controller translates stick inputs from the human pilot to motor inputs and can also be used for off-the-shelf GPS flight.

Manual pipeline The manual pipeline is almost identical to the ones used in manually controlled UAVs, with the only difference being the autonomy switch. The pipeline consists of a standard wirelessly connected transmitter/receiver pair. The servo signals are passed from the receiver through the autonomy switch to the flight controller. The manual to autonomous switching signal is controlled by a switch on the transmitter.

There is minimal software interference, and the software that is used in this process is the same software used in manual flight of a regular RC controlled UAV.

Autonomous pipeline The autonomous pipeline is dedicated to autonomous flight, based on on-board real-time processing of sensor data. Sensor fusion is used for interpreting surroundings and controlling the UAV. All sensor information is processed on the on-board processor board. The board can have several cameras connected to it, each possibly pointing in a different direction. The sensors connected to the microcontroller can provide additional information, which is sent over the USB bus to the processor board.

The software on the processor board processes the sensor information and translates it to commands which are sent over the USB bus to the microcontroller. The microcontroller translates these commands into standard servo pulse-width modulated (PWM) signals. These servo signals are sent to the flight controller through the autonomy switch.

Autonomy switch The autonomy switch provides a 'galvanic' separation between autonomous and manual flight. It gives a safe and instant method of taking control and is not controlled by the autonomous process. This gives the Twirre architecture a significant safety advantage compared to other architectures [9]. An additional virtue of this interface is that the UAV can be *partially* autonomous. For example, only the pitch of the UAV could run on the autonomous pipeline, while the remaining axes are manually controlled.

Cascade control system As follows from the distinction between low- and high-level control, Twirre utilizes a cascade control system. As mentioned before, for the low level a commodity flight controller is used. At the high level a control system is implemented in the software running on the processor board, which simulates the stick input from the human pilot. In autonomous mode, the mission specific software will determine the set-points for the high-level controller. By splitting up the control system in two layers it is convenient to change or update to a new or better flight controller.

Figure 2: The hexacopter. The diameter of the frame is 62 cm, or 85 cm including the protection ring.

3.2 Components

The specific implementation of the Twirre architecture described in this paper is built around a standard DJI Flame Wheel 550 (F550) hexacopter which consists of a frame and six motors, electronic speed controllers (ESC) and propellers. For the flight controller a NAZA-M V2 flight controller is used. This platform is chosen because of its modular design and the ability to use GPS-enabled flights. The hexacopter is shown in Figure 2.

The processor board consists of a Commell LS-37B mainboard with an Intel Core i7 3820QM processor with a clock speed of 2.7 GHz and a maximum turbo frequency of 3.8 GHz and 4 GB of DDR3 RAM. The processor board is running a standard Ubuntu 13.10 distribution. This high-end processing board makes it possible to perform on-board image analysis for autonomous flight.

Two industrial machine vision cameras are connected to the processor board. For the front camera an IDS UI-1241LE-C-HQ is used in combination with a 12 mm lens, whereas for the bottom camera an IDS UI-1221LE-M-GL is used in combination with an 8 mm lens. Industrial cameras are used because they have a global shutter. In a survey only consumer webcams using a rolling shutter where found, which are not suitable for image analysis applications with motion. To further stabilize the images the front camera is mounted on a servo controlled gimbal. This is shown in Figure 3.

The microcontroller board used is a MicroWii Flight Controller based on an ATmega32U4 microcontroller. This board is chosen mainly because it has all the necessary digital and analog in- and outputs. In this specific implementation it is not used as a flight controller, but it has a gyro / accelerometer (MPU-6050), a magnetometer (HMC-5883L) and a barometer (MS5611-01BA03) which can be useful during autonomous flight. An ultrasonic sensor (SRF08) is connected to the microcontroller for accurate altitude measure-

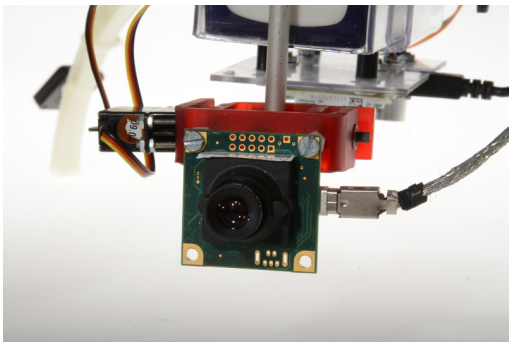

Figure 3: The IDS camera mounted on a servo controlled gimbal.

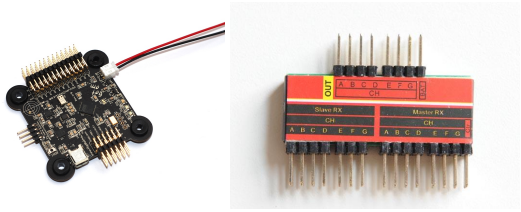

Figure 4: The MicroWii and the dual receiver controller.

ments.

The autonomy switch is implemented with a commodity $15 RC dual receiver controller that is designed for training RC pilots. In its intended use, two receivers are connected to the input side, and the outputs are connected to the servos or flight controller. Both the student-pilot and the instructor have their own transmitter and receiver. The instructor can, with the use of a switch on his transmitter, switch between the receiver that is active. The used MicroWii and the dual receiver controller are shown in Figure 4.

For the transmitter and receiver a Graupner MX-12 and a Graupner GR-12, both operating in the 2.4 GHz band, are used. The dual receiver controller is manufactured by Assan.

VisionLab[2] is used as image processing library.

4 METHODS

Now that the components of the Twirre architecture have been introduced, it is time to look at the system software parts. The first subsection describes the protocol software on the microcontroller. After that, the second subsection introduces a test application that runs on the processor board, followed by subsections describing the test application in more detail.

4.1 Protocol Software

The protocol software has two functions:

1. Interpret commands received from the USB bus, convert these commands to standard servo PWM signals and send them through the autonomy switch to the flight controller.
2. Read auxiliary sensor information, like gyro, barometer, etc. and send it over the USB to the processor board.

[2]http://www.vdlmv.nl

Figure 5: The POI being used in the test application.

4.2 Test application

In order to test the Twirre hardware architecture and the protocol software a basic UAV application is defined. In this application, the mission of the UAV is to search for an artificial POI, fly towards it and to face it. The POI being used is shown in Figure 5. It consists of a piece of A3 paper with a printed image of a solid black rectangular target with a white number inside.

There are many similarities between an autonomous UAV and UGV (Unmanned Ground Vehicle) performing this task. The main difference is when an UGV stands still, it actually stands still, whereas an UAV has the tendency to drift. The autonomous UAV flight discussed in this paper behaves like an UGV when searching for a POI. The test application software is a combination of the mission logic, input sensor data processing and the high-level control system.

4.3 Cascade PID controllers

The Twirre architecture uses cascade PID controllers to control the UAV. The low control level is embedded in the flight controller. The accelerometer, gyro and pressure sensor are used to keep the UAV leveled and at a constant height. When GPS can be used, the flight controller also controls the position of the UAV. These low-level controllers are commodity products. The high-level controllers are responsible for controlling the intelligent decision making of the UAV. In manually controlled UAVs high-level control is done by a human pilot.

The set-points for the high-level controller are determined by analyzing the surroundings and selecting a POI using the front camera and the processor board, and fly towards it. If there is no POI seen with the front camera, the bottom camera is used to keep a fixed position by estimating the drift using image analysis. In this case the yaw is used to rotate the UAV and scan the environment for new points of interest. The choice of bottom POI is made autonomously. The key is to always have a point of interest, whether seen from the front camera or from the bottom camera.

The Twirre architecture allows the UAV to be tested by mapping the pitch, yaw, roll and throttle to any combination of autonomous and manual control. This way individual, high-level controllers can be tested or several controllers can be tested simultaneously. This allows for detailed analysis of the autonomous behavior of the UAV. The autonomy switch

http://www.imavs.org/

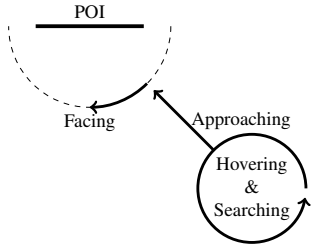

Figure 6: Top view of state transitions.

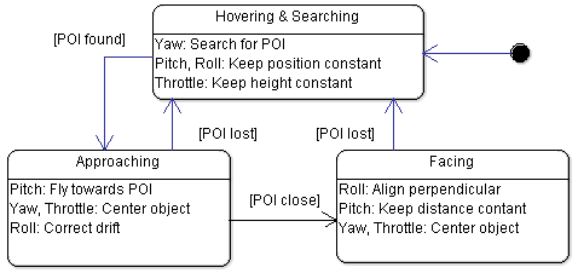

Figure 7: States of the high-level controller.

of the Twirre architecture can always be used to allow instant mapping to full manual control.

4.4 Controlling real-world parameters

Instead of focusing solely on image data, the Twirre architecture controls camera-independent, real-world parameters. So instead of using set-point deviations in pixels only, the platform relates these to real-world distances and angles, in millimeters and radians respectively. The advantage of this approach is that it allows for easy replacement and upgrading of cameras and lenses. Controllers are independent of the cameras and lenses being used. The PID controllers do not need to be re-tuned when a different camera or lens is mounted.

4.5 UAV states

For autonomous control in the test application three states are identified: *Hovering & Searching*, *Approaching* and *Facing*. The state determines how the controllers are connected to the UAV axes and which calculations are executed to investigate points of interest. A top view of the states is shown in Figure 6, and the corresponding state diagram is shown in Figure 7.

After taking off autonomously, the UAV enters the *Hovering & Searching* state. In this state the position of the UAV needs to stay fixed relative to the ground, while it rotates about its z-axis using yaw to scan the surroundings for POIs with the front camera. The drift from the hovering point is calculated by analyzing the relative translation using the bottom camera. The relative translation error is corrected by the pitch and roll axis of the UAV. The throttle is used to keep the height constant.

Whenever a POI is located with the front camera, the UAV is set to the *Approaching* state. In this state the UAV estimates the 3D position of the POI and navigates towards it using the pitch. The yaw and throttle are used to keep the POI centered in the image. The combination of pitch and yaw tends to make the UAV drift laterally, in the direction opposite to that of the yaw. A small roll correction proportional to the amount of yaw is used to compensate this.

If the UAV is close enough to measure the orientation of the POI, the *Facing* state is entered which centers and aligns the UAV perpendicular to it. The roll is used to align the UAV, the pitch is used to keep the distance constant, and the yaw and throttle are used to keep it centered. When the POI is lost, the state is restored to *Hovering & Searching*.

The remainder of this section describes the theory and ideas behind the control of each of the states.

4.6 Hovering & Searching

During the *Hovering & Searching* state, dense optical flow using the Farnebäck algorithm [14] is used. This algorithm estimates the drift of the UAV by analyzing two consecutive frames and estimating the translation and rotation using a pixel descriptor based on polynomial expansion. A constant altitude is maintained using the ultrasonic sensor.

4.7 Approaching

When the POI is found in the *Hovering & Searching* state, the *Approaching* state is entered. During this state the following tasks need to be completed:

1. Locate the POI with sub-pixel precision using the front camera and image analysis
2. Calculate the horizontal, vertical and distance deviations of the UAV with respect to the POI
3. Control these deviations to zero using yaw, throttle and pitch respectively

For the rectangular POI, the horizontal and vertical deviations are calculated as follows. The height of the target in the real world is known. In an image taken by the UAV the rectangle has a certain pixel height. With this it can be calculated how many real-world millimeters correspond to one pixel in the image:

$$\text{Scale factor} = \frac{\text{Real rectangle height (mm)}}{\text{Pixel rectangle height (px)}} \quad . \qquad (1)$$

The horizontal and vertical deviations in pixels can be found by comparing the center of the image to that of the POI in the image. Using the scale factor, these pixel deviations can be converted to deviations in millimeters.

The distance can be calculated using the equation for the magnification M in optics, using the thin lens model:

$$M = \frac{h_{object}}{h_{image}} = \frac{d_{object}}{d_{image}} \quad , \qquad (2)$$

http://www.imavs.org/

in which h_* and d_* are heights and distances in millimeters. The distance that should be calculated is d_{object}. Rewriting then gives:

$$d = d_{object} = d_{image} \frac{h_{object}}{h_{image}}$$

$$= f \frac{\text{Rectangle height (mm)}}{\text{Rectangle height (px)} \times \text{Pixel size (mm/px)}} \quad . \quad (3)$$

The effective focal length f and pixel size can be deduced from the specifications of the lens and camera. This distance is controlled to the set-point of the *Approaching* state's distance controller, which is equal to the set-point of the distance controller in the *Facing* state. As soon as the set-point is reached, the UAV transitions into the *Facing* state.

4.8 Facing

The *Facing* state does the same as the *Approaching* state, plus the correction of the perspective angle between the UAV and the POI. This angle can only be calculated reliably when the UAV is close to the POI and the POI is centered in the image. A schematic top view of the situation which uses the pinhole camera model is shown in Figure 8.

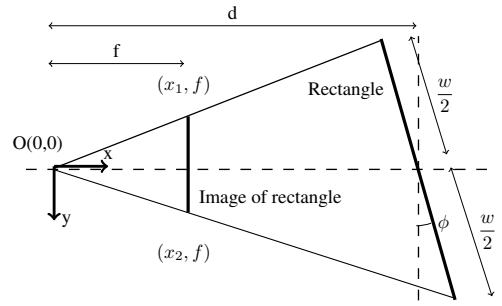

Figure 8: Top view of the camera at point O with focal length f facing the rectangle at an angle ϕ.

The camera with focal length f is located at point O. Depending on the perspective angle ϕ between the UAV and the POI, the ratio between the part of the image of the POI to the left of the center of the image (x_1) and the right (x_2) will change. The width of the real-world rectangle is given by w. From geometry it follows that ϕ is given by:

$$\tan\phi = \frac{1}{2}\left(\frac{f}{x_1} + \frac{f}{x_2}\right) \quad . \quad (4)$$

The angle is controlled to zero in order to directly face the POI, whereas the distance is controlled to a chosen set-point in front of it according to Equation 3, as explained before.

The theory described above assumes that the POI is not rotated about its horizontal axis. However, it can be extended to take this rotation into account.

5 EXPERIMENTS

A test application to test the Twirre architecture with autonomous flight has been developed. The test setting is shown in Figure 9. This section describes the tests that have been performed.

Figure 9: Experimental setting used during experiments.

5.1 Autonomy switch

The autonomy switch plays a central role in safety during the experiments. If a controller results in unstable behavior, the pilot can instantaneously take over the control to correct the movement of the UAV. The reliability of the switch is researched in a qualitative way, as well as its switching speed.

5.2 Hovering & Searching

The hovering is tested by taking off autonomously, setting the hovering set-point and letting the UAV hover autonomously at that point. The set-point of the height is at 1 meter above the ground. The hovering performance is measured by calculating the average longitudinal, lateral and height deviation during flight, along with their standard deviation and maximum value.

5.3 Approaching and Facing

During the *Approaching* and *Facing* experiments, autonomous control of every axis is tuned separately. So for the *Approaching* state, the yaw, throttle and pitch are tuned separately first. The PID gains are improved by looking at graphs of the UAV's response, showing deviations and steering inputs over time. After that, the controllers are combined, followed by fine-tuning in order to improve the combined control. The states are tested with a distance set-point of 3 meters from the POI.

6 RESULTS

The results obtained from the experiments explained above are shown in this section.

6.1 Autonomy switch

The autonomy switch was used extensively during the experiments. It was found perfectly reliable, as it worked correctly and instantly every time it was used.

http://www.imavs.org/

Table 1: Hovering accuracy measurements.

Deviation	Average (mm)	Std dev (mm)	Max (mm)
Longitudinal	-7.7	19	40
Lateral	9.2	21	44
Height	-7.4	8.6	30

6.2 *Hovering & Searching*

The hovering controllers result in the accuracy shown in Table 1. The numbers have been taken over nearly 600 frames at 40 FPS. It can be concluded that the hovering is accurate, with little deviations from the set-point.

6.3 *Approaching and Facing*

The resulting distance response during the *Approaching* state is shown in Figure 10. It shows the distance accurately converging to the set-point of 3 meters from the POI.

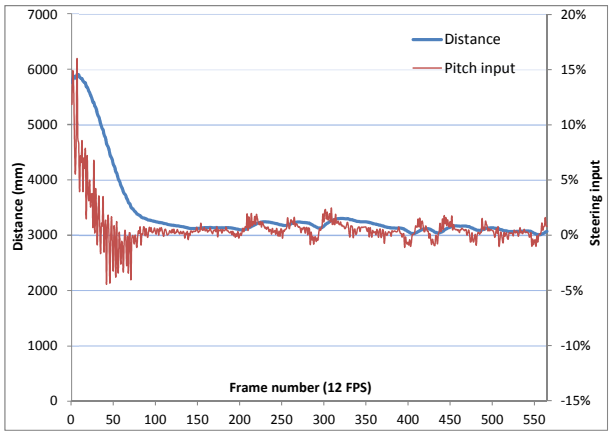

Figure 10: Distance control during the *Approaching* state.

The perspective angle deviation over time along with the roll steering inputs during the *Facing* state is shown in Figure 11. It shows the angle converging to zero.

Combining the *Approaching* and *Facing* state gives the performance shown in Figure 12. The *Approaching* state is active until frame 80 as can be seen in the perspective angle, which is in that case assumed to be zero because it cannot be measured reliably. For the state transition a distance margin of 0.4 meters around the set-point is used. As soon as the UAV enters this margin and the POI is centered, it transitions to the *Facing* state. This happens at frame 80, after which the perspective angle is no longer assumed zero. It can be seen that this angle is also controlled to zero. The graph shows accurate performance of controlling all four movements (roll, pitch, yaw and throttle) autonomously at the same time.

7 CONCLUSION

The Twirre architecture for UAVs consists of low-cost, easily interchangeable commodity hardware components. This

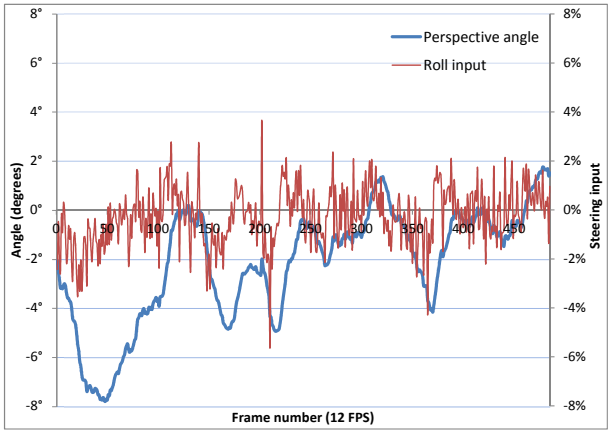

Figure 11: Perspective angle control during the *Facing* state.

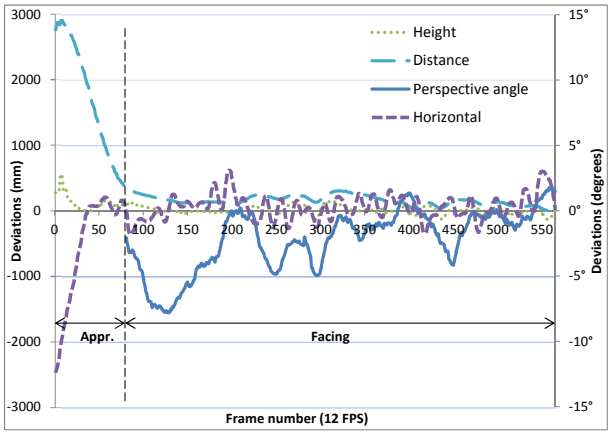

Figure 12: Deviations over time while performing the *Approaching* and *Facing* states after each other.

architecture has been used to implement a hexacopter using standard RC hobby components.

In the process of building and testing the hexacopter multiple components have been exchanged and were upgraded, while other components remained fixed. This was made easy because of the modular design of the Twirre architecture.

The autonomy switch is used to instantly switch back to manual mode without interference from the autonomous controller. Tests confirm that this is a reliable method for switching between manual and autonomous control.

A test application to demonstrate autonomous flight in a GPS-deprived environment is developed. Several experiments for autonomous flight towards a point of interest have been performed. The Twirre architecture allows for separate tuning of the high-level controllers. Experiments show that the tuned controllers show good performance in approaching and facing an object.

http://www.imavs.org/

8 FUTURE WORK

In the test application the software for the mission logic, the sensor data processing and the high-level control system are intertwined. Future work is to design and build reusable, non application specific software components, from which autonomous applications can be built. The goal is to separate the application specific software parts from the architecture, which makes the Twirre architecture more versatile for multiple applications. Future work will focus on the software design for the high-level control. This includes extending the state machine.

The current platform will also be combined with other sensors, like GPS, 3D sensors/cameras and multiple ultrasonic sensors. This will extend the platform to a multi-sensor fusion for robust autonomous flight in indoor and outdoor environments. The bottom camera will be put on a new downward-facing gimbal in order to have accurate optical flow measurements when moving over large distances. This will make it possible to implement new autonomous states like homing and landing.

9 REFERENCES

[1] Micropilot. *MP2x28 Family of UAV Autopilots*.

[2] Cloud Cap Technology. *Cloud Cap Technology Piccolo SL*.

[3] DJI. *A2 Flight Control System User Manual*, 1.14 edition, 02 2014.

[4] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft mav. In *2014 IEEE International Conference on Robotics & Automation (ICRA)*, 05 2014.

[5] G CHE de Croon, HW Ho, C De Wagter, E Van Kampen, B Remes, and QP Chu. Optic-flow based slope estimation for autonomous landing. *International Journal of Micro Air Vehicles*, 5(4):287–298, 2013.

[6] Laurie N Bose and Arthur G Richards. Mav belief space planning in 3d environments with visual bearing observations. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, 2013.

[7] Bart Remes, Dino Hensen, Freek van Tienen, Christophe de Wagter, Erik van der Horst, and Guido de Croon. Paparazzi: how to make a swarm of parrot ar drones fly autonomously based on gps. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013), Toulouse, France*, 2013.

[8] Andrew Nolan, Daniel Serrano, Aura Hernandez Sabaté, Daniel Ponsa Mussarra, and Antonio M López Pena. Obstacle mapping module for quadrotors on outdoor search and rescue operations. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013), Toulouse, France*, 2013.

[9] Balazs Gati. Open source autopilot for academic research the paparazzi system. In *roceeding of the American Control Conference 2013*, 2013.

[10] Jan Bolting, Francois Defaÿ, and Jean-Marc Moschetta. Differential gps for small uas using consumer-grade single-frequency receivers. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013)*, 2013.

[11] Jose Luis Sanchez-Lopez, Jesus Pestana, Paloma de la Puente, and Pascual Campoy. Visual quadrotor swarm for imav 2013 indoor competition. In *ROBOT2013: First Iberian Robotics Conference*, pages 55–63. Springer International Publishing, 2014.

[12] Kirill E. Shilov, Vladimir V. Afanasyev, and Pavel A. Samsonov. Vision-based navigation solution for autonomous indoor obstacle avoidance flight. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013), Toulouse, France*, 2013.

[13] Ming Liu, Cédric Pradalier, François Pomerleau, and Roland Siegwart. Scale-only visual homing from an omnidirectional camera. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3944–3949. IEEE, 2012.

[14] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

[15] G. Klein; D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[16] Blosch M., Weiss S., D. Scaramuzza, and Siegwart R. Vision based mav navigation in unknown and unstructured environments. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2010.

[17] Benjamin Ranft, Jean-Luc Dugelay, and Ludovic Apvrille. 3d perception for autonomous navigation of a low-cost mav using minimal landmarks. In *International Micro Air Vehicle Conference and Flight Competition (IMAV2013), Toulouse, France*, 2013.

http://www.imavs.org/